CSC 282 - Fall 2012

http://www.cs.rochester.edu/~stefanko/Teaching/12CS282/

1. (20 POINTS) We will call a sequence $b_1, \ldots, b_m$ **barely changing** if neighboring numbers differ by at most 73 (that is, $(\forall i \in \{1, \ldots, m-1\})|b_i - b_{i+1}| \leq 73$).

   Consider the following problem:
   INPUT: A sequence $a_1, \ldots, a_n$.
   OUTPUT: The length of the longest barely changing subsequence of $a_1, \ldots, a_n$.

   We would like to solve the problem using dynamic programming. Let

   $P[i]$ = the length of the longest barely changing subsequence of $a_1, \ldots, a_i$ that ends with a_i.

   Give an expression (or a piece of code) that gives an efficient way of computing $P[i]$ (assume $i \geq 2$).

2. (20 POINTS) We are given an $m \times n$ matrix A filled with numbers. A **monotone path** starts at the top-left square and in each step moves either down or right. (For example if $m = 2$ and $n = 3$ then there are 3 monotone paths that end at the bottom-right square: right-right-down (visiting squares $(1,1)$, $(1,2)$, $(1,3)$, $(2,3)$), right-down-right (visiting squares $(1,1)$,$(1,2)$,$(2,2)$,$(2,3)$), and down-right-right (visiting squares $(1,1)$,$(2,1)$,$(2,2)$,$(2,3)$).) The **value** of a monotone path P is the sum of the entries of A in the squares visited by P.

We want to find the largest value of a monotone path that ends at the bottom-right square. We are going to solve the problem using dynamic programming. Let

$$T[i,j] = \text{the largest value of a monotone path that ends at square } (i,j)$$

Give an expression (or a piece of code) that gives an efficient way of computing $T[i,j]$ (assume $i \geq 2$ and $j \geq 2$):

3. (20 POINTS) A **shuffle** of two strings A, B is formed by interspersing the characters into a new string, keeping the characters of A and B in the same order (for example, '**s**ev**e**r**al**' is a shuffle of 'seal' and 'evr'). Given three strings $A = a_1 \dots a_n$, $B = b_1 \dots b_m$, and $C = c_1 \dots c_{m+n}$, we would like to check whether C is a shuffle of A and B.

   Consider the following proposed algorithm for to the problem. We compute a table $T[0..n, 0..m]$ using the following rules

   - (for all $i \in \{0, \dots, n\}$) $T[i, 0]$ is true if and only if $a_1 \dots a_i = c_1 \dots c_i$,
   - (for all $j \in \{0, \dots, m\}$) $T[0, j]$ is true if and only if $b_1 \dots b_j = c_1 \dots c_j$, and
   - for other i, j the entry $T[i, j]$ is computed using the following update

   $$T[i, j] = (c_{i+j} = a_i \text{ or } c_{i+j} = b_j) \text{ and } (T[i-1, j] \text{ or } T[i, j-1]).$$

   Give an example of two stings A, B for which the algorithm produces wrong answer. Compute all the entries in the table T for your example (suggestion: keep your example small—that will save you work filing the table T).

4. (20 POINTS) Let $a_1, \ldots, a_n$ be a sequence of integers. We would like to split the sequence into segments such that for each segment the first and the last number are the same. For example, sequence $1, 3, 2, 1, 4, 1, 5, 6, 7, 8, 4, 8, 9$ can be split into 4 segments: $1, 3, 2, 1$ and $4, 1, 5, 6, 7, 8, 4$ and 8 and 9; or into 6 segments $1, 3, 2, 1, 4, 1$ and 5 and 6 and 7 and $8, 4, 8$ and 9 (there are other ways of splitting).

   We want to find the splitting that has the minimum number of segments (thus for the example above the first splitting into 4 segments is preferred). We are going to solve the problem using dynamic programming. We will compute a table $T[0...n]$ where $T[i]$ is the minimum number of segments to split $a_1, \ldots, a_i$. Give an expression (or a piece of code) for $T[i]$ in terms of $a_1, \ldots, a_i$ and $T[0], T[1], \ldots, T[i-1]$.